A Gradient-Based Implicit Blend

Olivier Gourmel, Loic Barthe, Marie-Paule Cani, Brian Wyvill, Adrien Bernhardt, Mathias Paulin, Herbert Grasberger

A Gradient-Based Implicit Blend Preprint

OLIVIER GOURMEL
IRIT, Université de Toulouse, CNRS, France
and
LOIC BARTHE
IRIT, Université de Toulouse, CNRS, France
and
MARIE-PAULE CANI
Laboratoire Jean Kuntzmann, Grenoble Universités, CNRS, INRIA Grenoble, France
and
BRIAN WYVILL
Department of Computer Science, University of Victoria, Canada
and
ADRIEN BERNHARDT
Laboratoire Jean Kuntzmann, Grenoble Universités, CNRS, INRIA Grenoble, France
and
MATHIAS PAULIN
IRIT, Université de Toulouse, CNRS, France
and
HERBERT GRASBERGER
Department of Computer Science, University of Victoria, Canada

We introduce a new family of binary composition operators that solves four major problems of constructive implicit modeling: suppressing bulges when two shapes merge, avoiding unwanted blending at a distance, ensuring that the resulting shape keeps the topology of the union, and enabling sharp details to be added without being blown up. The key idea is that field functions should not only be combined based on their values, but also *on their gradients*. We implement this idea through a family of C^∞ composition operators evaluated on the GPU for efficiency, and illustrate it by applications to constructive modeling and animation.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling

General Terms: Implicit Curves & Surfaces, Interactive Techniques

Additional Key Words and Phrases: Implicit surfaces, Blending, Constructive solid geometry (CSG), Geometric modeling, Surface-solid object representations

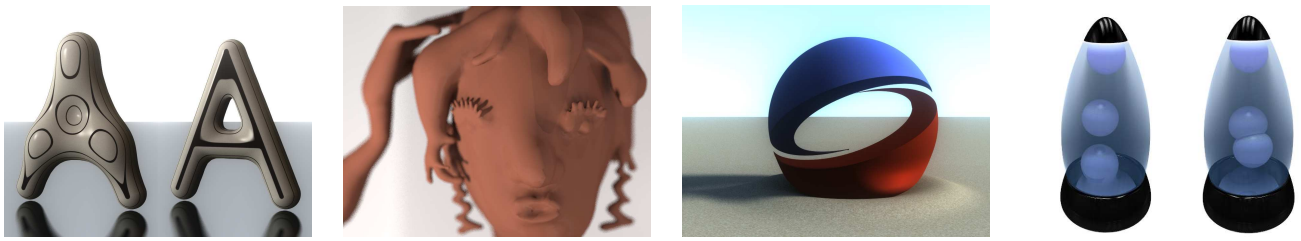
1. INTRODUCTION

Functionally-based implicit surfaces [Bloomenthal 1997b] were introduced in the eighties as promising alternatives to mesh-based and parametric representations. They opened the way to constructive modeling systems where different object components could be smoothly blended, and enabled the production of animations where fluid-like objects could change topology over time through successive merging and splitting. These effects were achieved by combining field functions using a composition operator. Surprisingly, after being enhanced in the nineties by the introduction of convolution surfaces [Bloomenthal 1997b] and of general construction trees [Wyvill et al. 1999], little advance was made on such operators in the last decade, leaving four major problems unsolved:

(1) *Bulging problem:* Implicit blending creates unwanted bulging. For instance, a shape generated by blending together some implicit cylinders defined in the same plane will necessarily be thicker where the cylinders join, as shown in Figure 1(a-left) where the 'A' is composed of three blended cylinders. The expected result is the 'A' on the right.

(2) *Locality problem:* Implicit models typically blend at a distance. This is a major problem in modeling applications, where preventing the blend between at least some parts of the models (e.g. between the hand and the head of the character in Figure 1(b)) is necessary. The issue is serious in animation too, where pieces of soft material should be allowed to deform each other and eventually blend, but only where they collide.

(3) *Absorption problem:* Sharp and thin details are smoothed and they blow up when blended into larger implicit primitives, since they are totally included into the support region of the latter. This problem prevents the creation of thin details such as the eyelashes in Figure 1(b). This is why implicit surfaces have the reputation of only producing simple, blobby shapes.



(a) (b) (c) (d)

Fig. 1. (a) The unwanted bulge created by standard blending (left) is suppressed by our gradient-based method while preserving the desired topological genus (right). (b) We also prevent small details from blowing up and primitives from blending where they do not intersect. (c) Our generic framework allows the modeling of a wide range of composition operators such as "bulge in contact" and (d) blending (left) and "bulge in contact" (right) combined in the same animation to mimic a lavalamp.

(4) *Topology problem:* Lastly, although implicit modeling is an excellent way of generating arbitrary topologies, blending often produces material that will fill a hole that a designer intended. For example the center of the 'A' in Figure 1(a-left). The user has no easy way to control, or at least to predict, the resulting topology.

Solving the above problems is essential for both constructive modeling and animation. While individual solutions have been proposed, none of them solves all these four issues at once. Doing so efficiently opens the way to many applications, from sketch-based, interactive design of 3D shapes (Figure 1(b)) to the interactive animation of deformable solids (Figures 1(c) and 1(d)). We review the state of the art and the contributions before presenting the required technical background and our general solution to these four problems.

1.1 Previous work

Implicit surfaces are well known for their ability to combine shapes by simply composing the associated field functions. Composition is generally expressed as a binary operation, except for the n-ary operators; *max*, which results in a union and *sum*, which generates smooth blending between the input shapes [Sabin 1968; Ricci 1973; Blinn 1982]. Improving composition operators has been a major topic of research, and modeling techniques were enhanced by the introduction of *clean union* operators generating an exact union of the input shapes but with a smooth field everywhere else and of blending operators with local shape control [Pasko et al. 1995; Barthe et al. 2004]. Other operators were developed for animation applications, such as the generation of a contact surface surrounded by bulges when soft objects collide [Cani 1993].

The elimination of the *bulging problem* was only tackled in two seminal papers: [Rockwood 1989] proposed an operator, the super-elliptic blend, where the range of the blend is parameterized by the cosine of the angle between the field gradients. With this formulation, the blend smoothly transitions to a union where the composed objects' surfaces become collinear. Even though suppressing the unwanted bulges, this operator was C^0 only and increased both the *locality problem* and the *topology problem*, as reported in [Bloomenthal 1997a]. The latter discussed alternative solutions for the n-ary composition of skeleton-based primitives: the kernels or the skeletons of convolution surfaces were used to parameterize a switch from sum to max operators. In addition to not being generally applicable, the lack of smoothness of these early solutions limited their usability in the case of successive blends. Our work builds on [Rockwood 1989] since our new composition operators are smooth functions of the angle between field gradients.

Although identified early [Bloomenthal 1997b], the *locality problem* was only solved recently in a general setting [Pasko et al. 2005; Bernhardt et al. 2010]. The proposed solutions are based on an extra implicit primitive, the blending volume, whose field is used to interpolate between blending and clean union (such as those of Figure 2). [Bernhardt et al. 2010] automatically generates blending volumes around intersection curves and sets parameters such that small primitives do not blow up, providing a solution to the *absorption problem* as well, however this method has some drawbacks:

(1) only G^1 continuous field functions are produced leading to curvature artifacts illustrated with reflection lines in Figure 3(c).

(2) Computations involve sampling the intersection curve between the two input surfaces or extracting the curve from the intersection of the meshes used for display. This makes the solution costly and not scalable, especially in applications where many intersection curves would need to be generated at different resolutions, such as in a detailed model.

The new method presented here is a more general, scalable approach and is able to restrict blending to regions where the input shapes intersect. At the same time the topological genus of the resulting shape can be constrained to remain that of the union (*topology problem*), which none of the previous solutions could achieve.

Among recent uses of implicit surfaces, we can notice several experimental sketch based modeling systems [Savchenko et al. 1995; Karpenko et al. 2002; Alexe et al. 2005; Wyvill et al. 2005; Bernhardt et al. 2008; Brazil et al. 2010] for which our new composition method would be especially useful.

1.2 Contributions

We present a generic family of composition operators that brings a unified solution to the four major problems mentioned above. The key idea is to parametrize the blending by the angle between the gradients of the combined field functions, as suggested by Rockwood [1989]. This enables us to interpolate between a union and a blend in the same composition. Using these operators we generate smooth blending between some parts of input shapes and union elsewhere. Blends near intersections are localized without requiring any extra implicit primitive, which makes the method efficient and scalable. In contrast with early work the operators described

using our system are C^∞ continuous, which is a sufficient property to handle successive compositions.

Our blending technique preserves sharp details and gives enough control on the blend localization so that in general, the topology of the composed object can remain that of the union, which makes composition predictable and well suited to constructive modeling frameworks. Our operators can also be tuned to animate progressive separation of soft material while preventing disjoint parts from blending at a distance. They can also generate contact surfaces surrounded by bulges, possibly followed by progressive merging, when soft objects collide.

The method is generic among functionally-based implicit surfaces. Although it can be applied to discrete fields as well, where f is defined as some interpolation of sample values stored in a grid, we do not address level-set modeling frameworks since the composed field functions we output are not governed by differential equations.

This paper derives our solution for local implicit primitives, such as soft-objects and meta-balls. As explained in the next section, these surfaces are the most challenging to handle since the composed field has to maintain compact support. However, our framework also applies to globally supported primitives, as we illustrate in Section 5.1 with convolution surfaces. Section 2 presents the necessary background and we refer to [Bloomenthal 1997b] for a full description of implicit models.

2. BACKGROUND

Implicit surfaces: A functionally-based implicit surface S is the C-iso-surface of a *field function* $f : \mathbb{R}^3 \to \mathbb{R}$, defined as:

$$S = \{p \in \mathbb{R}^3 \mid f(p) = C\}. \tag{1}$$

We can identify two types of field functions: globally supported and locally supported field functions.

In fields of global support, implicit surfaces are defined as 0-iso-surfaces and the inside of the surface can be either defined by points p in space such that $f(p) < 0$ as in Hoffman et al. [1985] and Barthe et al. [2001] works, or by points p in space such that $f(p) > 0$ as in Pasko et al. [1995] works.

Local support field functions are commonly positive, decreasing functions of the distance, for instance, to a skeleton, that vanish at a distance R from this skeleton. A skeleton can be a point, a line segment, etc. In this case, the implicit surface is an iso-surface "around" the skeleton, e.g. a sphere if the skeleton is a point. The implicit surface is the 0.5-iso-surface, and we use the convention that $f(p) > 0.5$ defines the interior and f decreases outside an object.

Composing field functions: One of the strengths of implicit modeling is the ability to combine fields to form a new shape, using a composition operator as:

$$f = g(f_1, f_2) \tag{2}$$

where $g : \mathbb{R}^2 \to \mathbb{R}$ is a binary composition operator, f_1 and f_2 are field functions defining the combined implicit surfaces and f is the field function defining the resulting object. As we will see, different operators are to be developed for global and local field functions.

Blending: A standard operator for generating smooth blends is the sum [Blinn 1982]:

$$g(f_1, f_2) = f_1 + f_2. \tag{3}$$

Unfortunately, it does not provide any control on the shape of the resulting blend, and can only be applied to local support field functions or to convolution surfaces. Therefore, several binary blending operators were proposed over the years for field functions with global support [Bloomenthal 1997b]. The idea is to blend 0-iso-surfaces while meeting specific limit properties, continuity constraints or aspect of iso-curves [Hoffmann and Hopcroft 1985; Rockwood 1989; Barthe et al. 2001]. For instance, the superelliptic blend [Rockwood 1989] is defined as:

$$g(f_1, f_2) = 1 - \left[\frac{1 - f_1}{r_1}\right]_+^t - \left[\frac{1 - f_2}{r_2}\right]_+^t \tag{4}$$

where $[*]_+ = \max(0, *)$, t controls the shape of the blend, making it closer to the combined objects when it increases, r_1 and r_2 control the boundaries of the blend on each combined primitive. This operator exhibits some gradient discontinuities that can be avoided by using Pasko et al. blending set theoretic operator [1995] defined as:

$$g(f_1, f_2) = f_1 + f_2 - \sqrt{f_1^2 + f_2^2} + \frac{a_0}{1 + \left(\frac{f_1}{a_1}\right)^2 + \left(\frac{f_2}{a_2}\right)^2} \tag{5}$$

where a_0, a_1 and a_2 have an equivalent effect on the blend than, respectively, t, r_1 and r_2 in Equation 4. Better field variations and user control can be achieved using Barthe et al. functional blend [2001] defined as:

$$g(f_1, f_2) = \min(f_1, f_2) - H\left(|f_1 - f_2|\right) \tag{6}$$

where $H : \mathbb{R} \to \mathbb{R}$ is a piecewise cubic polynomial defining the shape of the blend from user selected control points.

Union: Another well known operator performs a union using the $\max$ function [Sabin 1968; Ricci 1973]:

$$g(f_1, f_2) = \max(f_1, f_2). \tag{7}$$

This operator generates C^0 continuous field functions that can cause creases on subsequent blends with additional surfaces.

Clean union: Clean union operators have been introduced by Pasko et al. [1995] (Equation 8) and improved by Barthe et al. [2003] for field functions with global support.

$$g(f_1, f_2) = f_1 + f_2 - \sqrt{f_1^2 + f_2^2} \tag{8}$$

They improve on the standard $\max$ operator whose sharp field is illustrated in Figure 2(a-bottom) by performing a union of the surfaces while providing a smooth field function everywhere else (Figure 2(b-bottom)). In a sense, a clean union operator is a union operator on the combined surfaces and a blending operator of their

field functions everywhere else. This allows an object built using such an operator to be later blended with another implicit primitive without introducing unwanted sharp edges into the blend [Pasko and Adzhiev 2004].

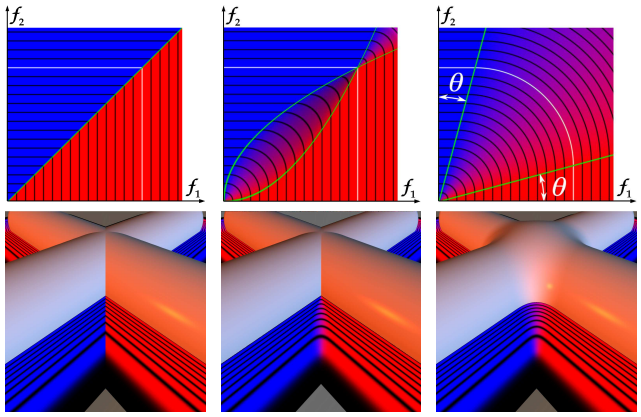


Fig. 2. Three compositions applied to a pair of cylindrical implicit primitives forming a cross: (a) standard union based on max, (b) clean union, (c) Barthe's blending operator parameterized by an opening angle θ. First row: plots of $g(f_1, f_2)$ where the axes are f_1 and f_2. Black lines: some iso-curves, white line: 0.5-iso-curve (surface of interest), background color: gradient direction with vertical (resp. horizontal) gradient in blue (resp in red), green lines: boundaries outside of which $g = \max(f_1, f_2)$. Second row: implicit surface $g(f_1, f_2) = 0.5$. Some iso-lines of g are depicted in a horizontal cutting plane.

Visual representation of binary operators: In their work, Hoffman and Hopcroft [1985] introduce an $\mathbb{R}^2$ space in which operators g are defined and plotted with the value of the field functions f_1 and f_2 respectively as abscissa and ordinate. In this space, vertical lines represent the iso-surfaces of the field function f_1 and the horizontal lines, those of the field function f_2. The top row of Figure 2 illustrates three operators plotted in this space in which the white line represents the object iso-surface. The composition operator $g = \max$ is represented as in Figure 2(a) while a clean union operator is represented as in Figure 2(b) and a blending operator that smoothly links iso-surfaces of the field function f_1 with those of the field function f_2 at the vicinity of their intersection (i.e. close to $f_1 = f_2$) is represented as in Figure 2(c).

Composing locally supported field functions: When using locally supported field functions, primitives are defined by 0.5-iso-surfaces and field functions uniformly equal zero outside the boundary of their support. Thus, composition operators have to be such that $g(0, f_2) = f_2$ (on and outside the support of f_1, g reproduces the field function f_2) and $g(f_1, 0) = f_1$ (on and outside the support of f_2, g reproduces the field function f_1). This also implies that $g(0, 0) = 0$. These properties make the definition of composition operators more subtle and tedious than for global supports. All these properties are fulfilled by the operators presented in Figure 2. For instance, the **clean union operator** presented in [Barthe et al. 2004], similar to the one depicted in Figure 2(b), is defined as:

$$g(f_1, f_2) = \begin{cases} \max(f_1, f_2) & \text{if } \left(f_1 \leq \frac{1}{2} \text{ and } \left(f_2 \geq \sqrt{\frac{f_1}{2}} \text{ or } f_2 \leq 2f_1^2\right)\right) \\ & \text{or } \left(f_1 > \frac{1}{2} \text{ and } \left(f_2 \leq \sqrt{\frac{f_1}{2}} \text{ or } f_2 \geq 2f_1^2\right)\right) \\ \{C : \hat{h}_C(f_1, f_2) = 1\} & \text{otherwise} \end{cases} \tag{9}$$

in which $\hat{h}_C$ is defined as:

$$\hat{h}_C(f_1, f_2) = \begin{cases} \frac{\sqrt{\left(f_1 - 2C^2\right)^2 + \left(f_2 - 2C^2\right)^2}}{C - 2C^2} & \text{if } f_1 \leq \frac{1}{2} \\ \frac{\sqrt{\left(f_1 - \sqrt{\frac{C}{2}}\right)^2 + \left(f_2 - \sqrt{\frac{C}{2}}\right)^2}}{C - \sqrt{\frac{C}{2}}} & \text{otherwise.} \end{cases} \tag{10}$$

An intricate closed form solution for the computation of C is given in [Barthe et al. 2004]. For **blending operators**, the sharpness of the blending shape can be controlled by a parameter t if the sum is replaced by [Ricci 1973]:

$$g(f_1, f_2) = \left(f_1^t + f_2^t\right)^{\frac{1}{t}}. \tag{11}$$

However, if some additional control is required on the blend boundary, as was achieved in the global support case through Equations 4, 5 and 6, then the blending operator proposed in [Barthe et al. 2004] needs to be used. This operator is illustrated in Figure 2(c) with $\theta = \theta_1 = \frac{\pi}{2} - \theta_2$ and defined as:

$$g(f_1, f_2) = \begin{cases} \max(f_1, f_2) & \text{if } f_2 \leq \tan(\theta_1) f_1 \text{ or } f_1 \leq \cot(\theta_2) f_2 \\ \{C : \tilde{h}_C(f_1, f_2) = 1\} & \text{otherwise} \end{cases} \tag{12}$$

in which $f_2 = \tan(\theta_1) f_1$ and $f_1 = \cot(\theta_2) f_2$ correspond to the green lines in Figure 2(c) (the $\max$ is used under the first line and on the left of the second). In between these two lines, $\tilde{h}_C$ is defined as:

$$\tilde{h}_C(f_1, f_2) = \frac{\left(f_1 - C.\cot(\theta_2)\right)^2}{\left(C - C.\cot(\theta_2)\right)^2} + \frac{\left(f_2 - C.\tan(\theta_1)\right)^2}{\left(C - C.\tan(\theta_1)\right)^2} \tag{13}$$

and θ_1, θ_2 control the boundaries of the blend on each combined primitive (as r_1, r_2 in Equation 4 and a_1, a_2 in Equation 5).

Opening angles to control the blend: Our solution is based on the work by Barthe et al. [2004] we already mentioned, in which the operator g, is designed to:

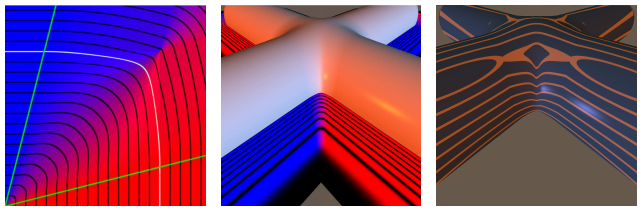
(1) give fine control over the resulting shape, as does the super-elliptic blend [Hoffmann and Hopcroft 1985; Hsu and Lee 2003],

(2) avoid the gradient discontinuities in the field function exhibited by displacement-blends [Rockwood 1989],

(3) handle field functions with local support.

The solution presented in Equation 12 uses the concept of an *opening angle* $\theta \in [0, \pi/4]$ (figure 2(c)) first introduced in [Barthe et al. 2003]. The blend is delimited by boundary lines (green lines in figure 2(c)). Inside these boundaries the iso-lines of $g(f_1, f_2)$ are arcs of ellipses defined in Equation 13 (as suggested in [Hoffmann and Hopcroft 1985]). Outside, no blend occurs and the resulting field function is defined as $f = g(f_1, f_2) = \max(f_1, f_2)$ (Equation 12) in order to exactly reproduce one of the fields of the input field functions. The boundary lines are controlled by varying the opening angle θ so that any intermediate blending (illustrated with $\theta = \pi/8$ in figure 2(c)) can be produced between a full blend when $\theta = 0$ and a union ($g = \max$) when $\theta = \pi/4$ (figure 2(a)). When $\theta \in [0, \pi/4]$, this operator is G^1 continuous and when $\theta = \pi/4$, it is C^0.

In practice, the opening angle can be set automatically from a point p, selected in object space $\mathbb{R}^3$, on one of the primitives where the user wants to switch between blending and union. For instance, suppose that the point $p(x, y, z)$ is selected on the 0.5-iso-surface of the field function f_1, i.e. $f_1(p) = 0.5$. Then, in the $\mathbb{R}^2$ space (figure 2(c) top) the opening angle θ is calculated with the values of the field functions f_1 and f_2 at the point p as $\theta = \arctan(f_2(p)/f_1(p)) = \arctan(2f_2(p))$. The point p can also be set automatically from the relative size of the composed primitives [Bernhardt et al. 2010].



(a) (b) (c)

Fig. 3. Bernhardt's operator interpolating between clean union and Barthe's blending. The distortions of the iso-curves of g (a) result in a G^1-only implicit surface (b) which still has a bulge on top and exhibits a poor curvature distribution, shown by depicting reflection lines (c).

Interpolating between clean union and blending: Operators interpolating between clean union and blending are able to perform both union and a smooth blend in the same composition. For instance, they enable the combined objects to blend where they are in contact while avoiding any shape deformation everywhere else. They have been introduced by Pasko et al. [2005] for globally supported field functions. Pasko's operator is defined by Equation 5 in which the coefficient a_0 varies: if $a_0 = 0$, the operator performs a clean union (as in equation 8), and the larger a_0, the larger the blending. Bernardt at al. [2010] proposed a such operator for locally supported field functions. This operator linearly interpolates between the clean union operator defined in Equation 9 and the blending operator defined in Equation 12. The limitations of these methods have been discussed in Section 1.1 and Bernhardt's operator is illustrated in Figure 3. Note that the clean union operator introduced in [Barthe et al. 2004] (Figure 2(b)) does not support the insertion of an opening angle that would allow the interpolation between a blend and a clean union.

3. GRADIENT-BASED COMPOSITION

Let f_1 and f_2 be the field functions of the implicit primitives to be composed and g be the composition operator we are looking for. As in [Rockwood 1989], we claim that g should not only depend on f_1 and f_2, but also on their gradients. Indeed, this appears as a pertinent option to smoothly blend intersecting shapes where they form a sharp angle, while avoiding bulges where their surfaces are already aligned. This section first presents the key features of our solution. We then explain the field continuity issues that motivate the introduction of a family of quasi-$\mathcal{C}^\infty$ operators in Section 4.

3.1 Controlling opening from angle between gradients

As we just mentioned, g should be able to generate smooth blending between selected parts of the input shapes and union elsewhere. Following [Barthe et al. 2004] we choose g with an opening angle parameter $\theta \in [0, \pi/4]$ (see Figure 2(c) and Section 2). θ controls the locality of the blend, i.e. the limit values in the (f_1, f_2) space out of which a union is computed.

The key feature of gradient-based blending is to define the opening angle θ as a function of the angle α between the field gradients at the query point p:

$$\theta = \theta(\alpha(p)) \text{ with } \alpha(p) = \arccos \frac{\nabla f_1(p) \cdot \nabla f_2(p)}{\|\nabla f_1(p)\| \|\nabla f_2(p)\|} \quad (14)$$

where α is the angle between the field gradients and the opening angle $\theta : [0, \pi] \to [0, \pi/4]$ is defined as a continuous function we call an *opening function*.

While a constant opening function θ applies the same type of composition everywhere along the input surfaces (such as smooth blending, shown in the first row of Figure 4), choosing opening functions that adequately tune the opening angle according to α allows us to seamlessly solve the four challenging problems we mentioned. Figure 4 shows how this is achieved; from left to right we have:

(a) The *bulging problem*: the 'T' in the top row shows the inflation produced by smooth blending where surfaces are aligned (α close to 0). To avoid this, the opening function should switch to union when $\alpha = 0$ (aligned gradients). This is done by choosing θ such that $\theta(0) = \pi/4$ (union for aligned gradients) and $\theta(\pi/2) = 0$ (blending for orthogonal gradients).

(b) This behavior also naturally prevents the *absorption problem* on small details since the inflation produced by the blend is gradually reduced as we come close to the "top" of sharp features (where the input surfaces are aligned).

(c) The *locality problem* occurs when two surfaces come close to each other and blend while they do not intersect, so when α is close to π. This is easily avoided by choosing θ such as $\theta(\pi) = \pi/4$ (union for opposing gradients) and $\theta(\pi/2) = 0$ (blending for orthogonal gradients).

(d) This setting also avoids the *topology problem*, i.e. the change of topological genus shown on top, since there are points inside each handle where the angle between gradients is π and which will therefore remain outside of the composed shape.

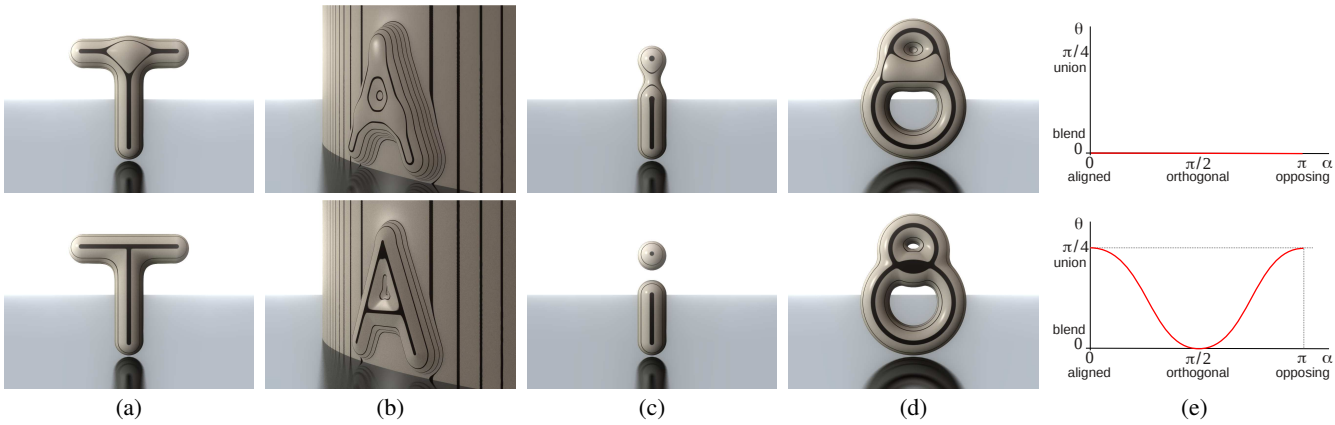


Fig. 4. Comparison between standard blending (first row) - equivalent to a null opening function - and our method (second row) with a opening function that smoothly switches from clean union when $\alpha = 0$ or $\alpha = \pi$ to blending when $\alpha = \pi/2$. Our solutions to: (a) unwanted bulge, (b) blow up of small details, (c) blending at distance, (d) topology modification. (e) Plot of $\theta(\alpha)$.

The opening functions used to generate these compositions are depicted in Figure 4(e). Note that the one on the second row, which we call the *Camel opening function* for its shape, solves all four challenging problems we mentioned. It will be fully described in Section 5.

3.2 Continuity issues

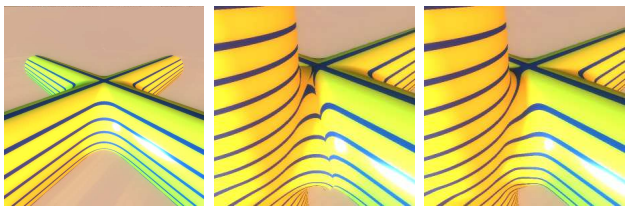


Fig. 5. (a) Two cylindrical implicit primitives are combined using Barthe's G^1 operator, where our gradient-based opening function is used to set the opening angle. The resulting shape still looks smooth. (b) A third primitive is added with the same blending operator. A C^0 sharp crease now appears. (c) Same scene using our new quasi-$\mathcal{C}^\infty$ blending operator.

Computing composition based on the gradients of the input fields having a finite level of continuity locally causes a loss of one order of continuity for the resulting field. This decrease cannot be avoided, but efforts can be made such that neither the opening function θ nor the composition operator g (based on it) introduce any extra source of discontinuity, which could cause artifacts in case of consecutive blends. In particular, Barthe's operator [Barthe et al. 2004], depicted in Figure 2(c), cannot be used in our framework since it is G^1 only and sets up a transition to a standard, C^0 only union. If the opening angle of this operator is set to a function θ of the field gradients, it generates a G^1 field function with the expected blend after the first composition (Figure 5(a)) and the loss of continuity after each composition results in a C^0 field function with a sharp edge in the second blend (Figure 5(b)) while a smooth blend as the one illustrated in (Figure 5(c)) was expected. Higher degree polynomials could be used to define the composition operator, but this would still limit the number of compositions that can be consecutively performed in the same region. Instead, our goal is to define $\mathcal{C}^\infty$ operators g based on $\mathcal{C}^\infty$ opening functions θ and to set them to control transition to a clean union (with a smooth field everywhere outside the surface of interest) when $\theta = \pi/4$.

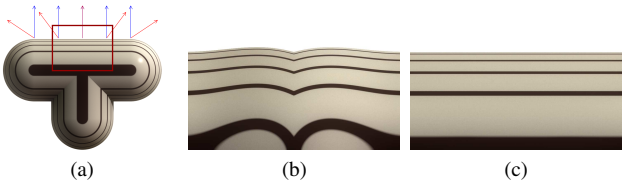
The constraint for g and θ to be $\mathcal{C}^\infty$ is in fact theoretical: in practice, we just need g and θ to approximate some $\mathcal{C}^\infty$ functions at a given precision. This has to be done without introducing oscillations in the field, since the latter would result in curvature artifacts. We say that a function verifying this looser constraint is quasi-$\mathcal{C}^\infty$ continuous. Relaxing our continuity constraint to quasi-$\mathcal{C}^\infty$ will enable us to efficiently store our operators as textures on the GPU and to evaluate them with hardware linear interpolation.

Section 4 introduces our opening functions and our new composition operators, which are applied to shape modeling and animation in Section 5. In these sections, the input fields are supposed to be singularity free, i.e. with smoothly varying, non-zero gradients vectors. The application of our framework to more general settings will be discussed in Section 6.

4. QUASI-$\mathcal{C}^\infty$ OPERATORS

4.1 Opening functions

In this section we describe a practical set of $\mathcal{C}^\infty$ functions for defining the opening functions θ that convert the angle between gradients into the opening angle of the composition operator. Based on the work of section 3 this family of functions should be flexible enough to implement various modes of conversion, including, but not restricted to those of Figure 4(e). Opening functions designed to model organic shapes and to animate contact situations will be shown in Section 5. The family of functions needed for θ should enable to set some key values and provide some slope control; without which, unwanted bumps may appear, depending on the slopes of the input fields around regions where a clean union is applied. See Figure 6.



(a) (b) (c)

Fig. 6. Role of the slope of θ in blending. The red (resp. blue) vectors are gradients of the horizontal (resp. vertical) primitive. (a) Union. (b) Blending with a sharp opening function ($w_0 = w_1 = 3$): unwanted bumps appear where the surfaces are almost aligned, since θ falls very quickly to 0 there. (c) Tuning the slope of θ according to the type of input primitives while keeping its other parameters unchanged solves the problem. Here, we set $w_0 = w_1 = 1$, a value pre-selected for soft objects.

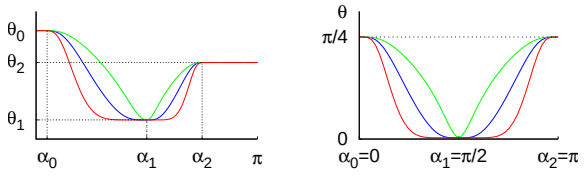
We therefore use a class of functions $\theta(\alpha), \alpha \in [0, \pi]$ defined from eight parameters $\alpha_0, \alpha_1, \alpha_2, \theta_0, \theta_1, \theta_2, w_0$ and w_1:

$$\theta(\alpha) = \begin{cases} \theta_0 & \alpha \leq \alpha_0 \\ \left(\kappa\left(\frac{\alpha-\alpha_1}{\alpha_0-\alpha_1}\right)\right)^{w_0}(\theta_0-\theta_1)+\theta_1 & \text{if } \alpha \in]\alpha_0, \alpha_1] \\ \left(\kappa\left(\frac{\alpha-\alpha_1}{\alpha_2-\alpha_1}\right)\right)^{w_1}(\theta_2-\theta_1)+\theta_1 & \text{if } \alpha \in]\alpha_1, \alpha_2[\\ \theta_2 & \text{otherwise} \end{cases} \quad (15)$$

where

$$\kappa(x) = 1 - \exp\left(1 - \frac{1}{1-\exp\left(1-\frac{1}{x}\right)}\right). \quad (16)$$

This function is controlled with eight parameters: three values of α_i ($i = 0, 1, 2$), where α_0 and α_2 are limits of the intervals $[0, \alpha_0]$ and $[\alpha_2, \pi]$ in which θ is set to be constant, and the three associated values for θ ($\theta_0 = \theta(\alpha_0)$, $\theta_1 = \theta(\alpha_1)$, $\theta_2 = \theta(\alpha_2)$) and two additional parameters w_0 and w_1 controlling the slope of the opening function in respectively $[\alpha_0, \alpha_1]$ and $[\alpha_1, \alpha_2]$. See Figure 7. In order to ensure C^∞ continuity, θ is piecewise defined from a function κ of class C^∞ that is flat at the boundaries of its support (i.e. all their successive derivatives equal zero).



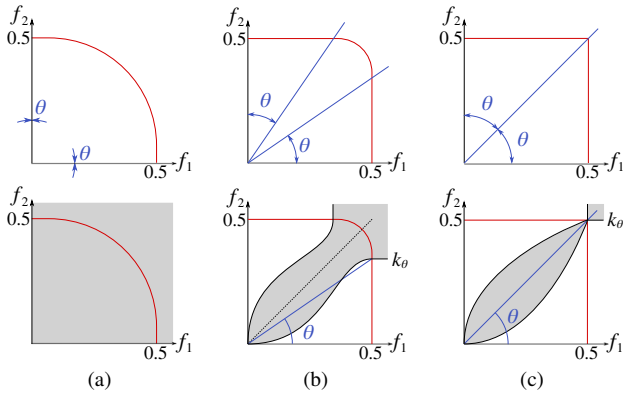
(a) (b)

Fig. 7. (a) The 8 parameters of a opening function θ with $w_0 = w_1$ respectively equal to 3, 1 and $1/3$ for the red, blue and green curves. (b) Three possible opening functions for the blend in Figure 6, using the same values for w_0 and w_1.

4.2 Composition operators

Let us now define another family of quasi-$\mathcal{C}^\infty$ functions for the composition operator g. This operator should produce smooth blends and also be able to generate other useful compositions, such as bulges when soft objects come in contact [Cani 1993]. Whatever the type of composition, the operator should be parameterized by the opening angle $\theta, \theta \in [0, \pi/4]$. The challenge is to define an operator that will fully apply the composition when $\theta = 0$ (opened operator, e.g. a smooth blend), and gradually switch to a clean union between the input shapes when $\theta = \frac{\pi}{4}$ (closed operator), while avoiding the limitations of Bernard's operator (Section 1.1 and Figure 3).

Following the construction of operators with an opening angle, our operator g uses the angle θ to define the width of the region where the composition is applied at the level of the iso-value (red iso-curve in Figure 8). We also use boundary curves k_θ bounding the region where the values of g modifies the field of the combined field functions f_1 and f_2 (light grey area in Figure 8). In this region, $g = \bar{g}$ and $g = \max$ outside. A critical contribution here is the special shape of our boundary curves k_θ when θ varies from 0 to $\pi/4$ (Figure 8(b)). This shape has been specifically designed to continuously interpolate between the vertical and horizontal boundary lines of a standard blend ($\theta = 0$, Figure 8(a)) and the smooth boundary curves intersecting at $f_1 = f_2 = 0.5$ of a clean union ($\theta = \pi/4$, Figure 8(c)).



(a) (b) (c)

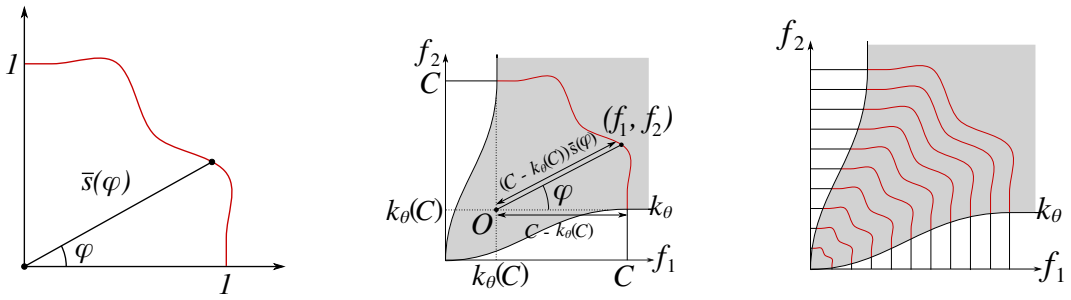
Fig. 8. Our operator g. First row: The opening angle θ defines the width of the region where composition (here, smooth blend) is applied, at the level of the iso-value (red iso-curve). Second row: The boundary curves k_θ bound the region where the values of g are computed with $\bar{g}$ (light grey area). A $\max$ operator is used outside. (a) Opened operator ($\theta = 0$). (b) Intermediate situation. (c) Closed operator ($\theta = \pi/4$).

We introduce a family of $\mathcal{C}^\infty$ boundary curves whose intermediate shape is shown in the second row of Figure 8 (b) and in Figure 9 (c). These boundary curves are defined by $\mathcal{C}^\infty$ functions $k_\theta(f)$, where f is either f_1 or f_2. We set $k_\theta(f) = \tan(\theta)/2$ inside of the implicit primitives, where $f \geq 0.5$. Outside the primitives, where $f < 0.5$ the expression of k_θ is:

$$k_\theta(f) = \frac{\tan(\theta)}{2}\left(\frac{4}{1+\tan(\theta)}\lambda_\theta(f)\right)^2, \quad (17)$$

with

$$\lambda_\theta(f) = \begin{cases} f & \text{if } f \leq \frac{\tan(\theta)}{2} \\ \frac{1-\tan(\theta)}{4}\phi(2\frac{2f-\tan(\theta)}{1-\tan(\theta)}) + \frac{\tan(\theta)}{2} & \text{otherwise} \end{cases} \quad (18)$$



(a) (b) (c)

Fig. 9. (a) A silhouette curve $\bar{s}(\varphi)$, (b) defines the shape of the C-iso-curves of $\bar{g}$ to produce (c) the corresponding operator g.

and ϕ is a function ensuring the C^∞ continuity of λ_θ. ϕ is computed by binary search at the required precision, from ϕ^{-1} defined as:

$$\phi^{-1}(x) = x + \frac{1}{e}\log\left(\log\left(\frac{1}{\nu(x)} + 1\right) + 1\right), \qquad (19)$$

where $e = \exp(1)$ and with

$$\nu(x) = \exp\left(\exp(e - ex) - 1\right) - 1. \qquad (20)$$

In practice, the computation of ϕ does not affect efficiency since our operators are pre-computed on the GPU (see Section 4.3). The value of g inside the region bounded by the *boundary curves* k_θ (in grey on Figures 8 and 9 (c)) is defined by an operator $\bar{g} : \mathbb{R}^2 \to \mathbb{R}$, which sets the type of composition we desire. The equation of our composition operator g is thus:

$$g(f_1, f_2) = \begin{cases} \max(f_1, f_2) & \text{if } f_1 \leq k_\theta(f_2) \text{ or } f_2 \leq k_\theta(f_1) \\ \bar{g}(f_1, f_2) & \text{otherwise} \end{cases} \qquad (21)$$

The function $\bar{g}$ is defined from a *silhouette curve* $\bar{s}(\varphi) : [0, \pi/2] \to \mathbb{R}$ describing the shape of the iso-curves of $\bar{g}$ as illustrated in Figure 9. For instance, an arc-of-a-circle is created using $\bar{s}(\varphi) = 1,\ \forall\varphi$. For given values of f_1 and f_2, the value of our operator $\bar{g}(f_1, f_2)$ is computed by solving the following equation in C:

$$\bar{g}(f_1, f_2) = \{C : \bar{h}_C(f_1, f_2) = 1\} \qquad (22)$$

with

$$\bar{h}_C(f_1, f_2) = \frac{\sqrt{(f_1 - k_\theta(C))^2 + (f_2 - k_\theta(C))^2}}{\bar{s}(\varphi)\,(C - k_\theta(C))} \qquad (23)$$

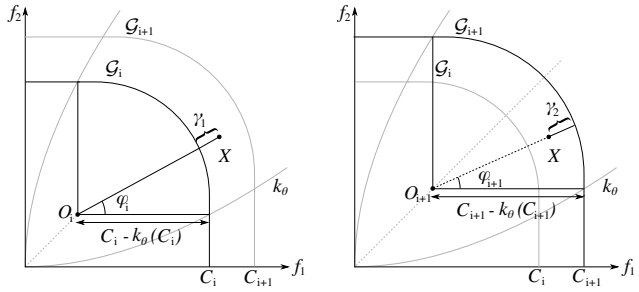
$$\text{and} \quad \varphi = \arctan\frac{f_2 - k_\theta(C)}{f_1 - k_\theta(C)} \qquad (24)$$

which expresses the fact that the iso-curves of $\bar{g}$ are scaled versions of $\bar{s}$. In practice, equation (23) does not always have a closed form solution. We thus precompute sampled values of g and interpolate them on the GPU, as detailed in Section 4.3.

Using any $\mathcal{C}^\infty$ silhouette curve $\bar{s}$, such that $\bar{s}(0) = \bar{s}(\pi/4) = 1$ and with flat derivatives at the level of the boundary curve (so that g's derivatives match those of the max function) is thus sufficient for getting a $\mathcal{C}^\infty$ operator g. To illustrate how different these silhouette curves can be and the variety of resulting compositions, Section 5 will give the equations of the curves we use for smooth blending and for a "bulge-in-contact" composition.

4.3 Implementation on the GPU

As discussed above, we use families of opening functions and of composition operators that have all the required smoothness properties, but that may not have any closed form expression. For efficiency, we developed a GPU implementation for these functions, which enables us to apply gradient-based compositions in real-time.



(a) (b)

Fig. 10. Computation of the interpolation parameter γ_1 (a) (respectively γ_2 (b)), defined as the distances from X to $\mathcal{G}_i$ (respectively from X to $\mathcal{G}_{i+1}$), used to compute $\bar{g}$ at a sample point X in Equations 25, 26 and 27.

The GPU implementation is based on the following pre-computations: the opening function which gives θ from the angle between gradients (equation 15) and its derivative $\frac{\partial\theta}{\partial\alpha}$ are stored as 1D textures. We also pre-compute values of the operator g and of its partial derivatives $\frac{\partial g}{\partial f_1}$, $\frac{\partial g}{\partial f_2}$, $\frac{\partial g}{\partial\theta}$ as functions of θ, f_1 and f_2 in a regular grid stored as a 3D texture. This is done efficiently using the following procedure: The values of g are computed by grid slice (f_1^i, f_2^j), $(i, j) \in [0; N-1]^2$, for each grid coordinate θ_k, $k \in [0; N-1]$ (N^3 being the size of the grid). In a slice, for each i, we set $C_i = f_1^i$ and we follow the C_i-iso-curve $\mathcal{G}_i$ of g and update the sample values as follows (see Figure 10):

—Outside of the boundary curves, i.e. for each j such that $f_2^j < k_\theta(f_1^i)$, $g = \max(f_1, f_2)$ (Equation 21), so we set $g(f_1^i, f_2^j) = C_i$ and $g(f_2^j, f_1^i) = C_i$.

—Inside of the boundary curves, for each sampling point $X(X_x, X_y)$ located in the bounding box of the silhouette curve $\bar{s}$ scaled by $(C_{i+1} - k_\theta(C_i))$ with the bottom-left corner in O_i, we compute γ_1 and γ_2 as in Equations (26) and (27). When $\gamma_1 \geq 0$ and $\gamma_2 \geq 0$, X is between $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$, i.e. the C_i-iso-curve and the C_{i+1}-iso-curve, and we compute $\bar{g}(X)$ by linearly interpolating C_i and C_{i+1} as in Equation (25).

$$\bar{g}(X) = \frac{\gamma_2 C_i + \gamma_1 C_{i+1}}{\gamma_1 + \gamma_2} \tag{25}$$

with

$$\gamma_1 = \|X - O_i\| - \bar{s}(\varphi_i)\,(C_i - k_\theta(C_i)) \tag{26}$$
$$\gamma_2 = \bar{s}(\varphi_{i+1})\,(C_{i+1} - k_\theta(C_{i+1})) - \|X - O_{i+1}\| \tag{27}$$

where

$$O_i = (k_\theta(C_i), k_\theta(C_i)) \quad \text{and} \quad \varphi_i = \arctan \frac{X_y - k_\theta(C_i)}{X_x - k_\theta(C_i)}. \tag{28}$$

Once all the values of g are computed, its partial derivatives are evaluated using finite differences and stored in other grids.

During field queries for a composed implicit surface, the values in these textures are linearly interpolated on the GPU: we first get θ from the gradients of the input fields and then $g(f_1, f_2)$ with this specific opening angle θ. Depending on the primitive, gradients are either computed using a closed form expression or finite differences. To efficiently apply successive compositions, the gradient of $g(f_1, f_2)$ is computed as:

$$\nabla g = \nabla f_1 \frac{\partial g}{\partial f_1} + \nabla f_2 \frac{\partial g}{\partial f_2} + \nabla \alpha \frac{\partial \theta}{\partial \alpha} \frac{\partial g}{\partial \theta} \tag{29}$$

A consequence of our GPU implementation is that the operators we use in practice are not exactly the $\mathcal{C}^\infty$ functions we developed, but quasi-$\mathcal{C}^\infty$ approximations of the latter: indeed, linear interpolation of values in a texture prevents unwanted oscillation and enables us to approximate the functions at any desired precision. We tested the method with 16^3, 32^3, 64^3 and 128^3 grids for g and its gradients. Figure 11 shows the improvement of reflection lines on a very close view of a composed primitive when texture resolution increases. Our current implementation uses 128^3 grids which is accurate enough to prevent any visual artifact, even after multiple compositions. Note that if a very high precision was required, e.g. for a manufacturing application, an off-line evaluation mechanism with bounded error could be set up, where binary search would be used to find the resolution needed to obtain a given error bound.

5. APPLICATIONS

In this section, we present applications of our generic composition operators to constructive modeling and animation. This leads us to define specific silhouette curves $\bar{s}$ and the associated opening functions θ.

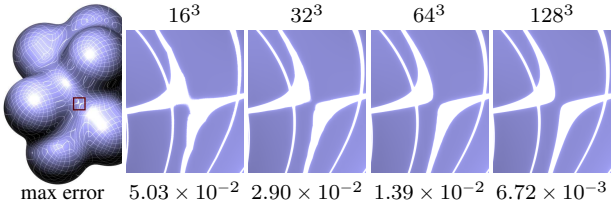


Fig. 11. Close-up on the surface obtained after three successive blends when g is stored into textures of different resolutions. The piecewise linear interpolation generates artifacts in reflections (here illustrating the continuity of the fifth derivatives) with the 16^3 and 32^3 grids that are not perceptible using higher resolutions. The third and fourth rows respectively give the max and mean approximation error provided by the tri-linear interpolation of the pre-computed values.

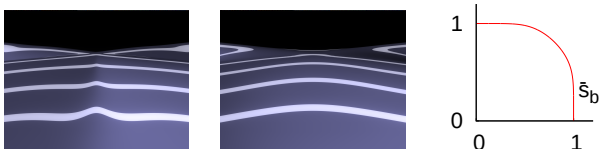


Fig. 12. Close view of the top of the cross formed by two blended cylindrical primitives: (a) using Bernhardt's blending; (b) using our operator g_b with a opening function preventing the unwanted bulge; (c) Plot of $\bar{s_b}$ from which g_b is derived.

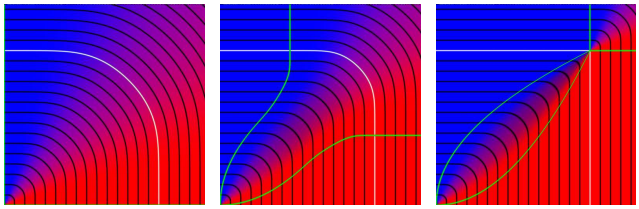


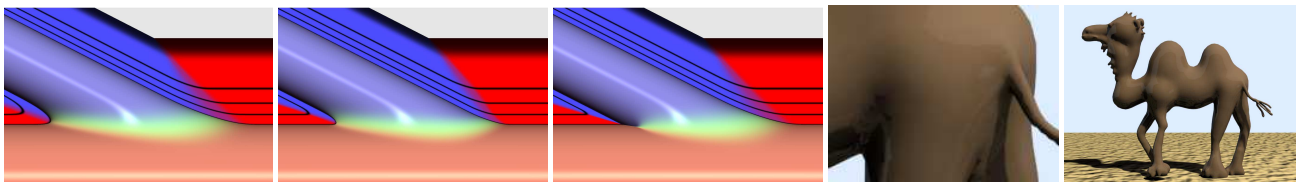
Fig. 13. Plots of g_b in the (f_1, f_2) space for different values of the opening angle θ. (a) Opened operator: $\theta = 0$. (b) $\theta = \frac{\pi}{8}$. Note the improved shape of the iso-curves compared to those of Figure 3. (c) Closed operator, resulting in a clean union: $\theta = \frac{\pi}{4}$.

5.1 Smooth blending for constructive modeling

Being able to smoothly blend arbitrary shapes is one of the most useful compositions in constructive modeling systems. To create a good blending operator $\bar{g_b}$, we need a $\mathcal{C}^\infty$ silhouette curve $\bar{s_b}$, which, in addition to meeting the value and derivative constraints listed at the end of Section 4.2 (values 0.5 and null derivatives of every order at the extremities), avoids curvature oscillations as much as possible, as illustrated in Figure 12.

Our solution plotted in Figure 12(c) is a symmetric silhouette curve whose curvature monotonically increases, with a maximum at $\varphi = \pi/4$, then decreases back to zero. We define $\bar{s_b}$ as follows:

$$\bar{s_b}(\varphi) = 1 - \frac{1}{e} \log\left(\log\left(\frac{1}{\nu(\varphi)} + 1\right) + 1\right) \tag{30}$$



Barthe's blending | "camel" blending | "organic" blending | "organic" blending in interactive modeling

Fig. 14. Comparison of Barthe's blending with our operator g_b, used either with the "camel" or with the "organic" opening function.

where ν is defined in Equation 20.

For our pre-computations of g on the GPU, we express this profile function in polar form, as $\bar{s_b}(\varphi)$, by sampling the curve at points $X_i = (x_i, \bar{s_b}(x_i))$ and pre-computing the value of φ_i at each X_i. To evaluate the function $\bar{s_b}(\varphi)$, we use binary search to find the interval $[\varphi_i; \varphi_{i+1}]$ containing φ, and then use linear interpolation. Although not exactly flat at the boundaries, this silhouette curve has zero derivatives in $\varphi = 0$ and $\varphi = \pi/2$ up to an error of $\epsilon = 10^{-5}$, which is accurate enough for our needs. This operator is illustrated in $\mathbb{R}^2$ space in Figure 13 for different values of θ. Its fairness leads to a clear improvement compared with Bernhart's blending solution as shown with reflection lines in Figures 12(a) and 12(b).

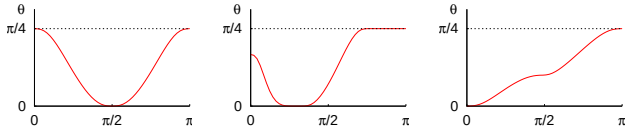
This blending operator g_b computed using $\bar{s_b}$ can be combined with different opening functions according to the need. Our applications to shape modeling lead us to design two specific examples of opening functions.

The first one, whose parameter values are given in the first row of Table I, is called the *camel opening function* because of the shape of the associated curve (see its plot in Figure 15(a)). It is symmetric and was especially designed to solve, for Wyvill's local-support soft-objects [Wyvill et al. 1986], all four problems mentioned in the introduction (Figure 4). In the case of CAGD applications, where two primitives join, this opening function provides smooth, predictable blending behavior while preventing bulges.

The second one, whose parameter values are given in the second row of Table I, is called the *organic opening function* (see its plot in Figure 15(b)). It was developed for an interactive modeling system dedicated to the modeling of organic shapes, using global-support convolution surfaces based on Cauchy kernel [Tai et al. 2004]. Here, being able to model both smooth and sharp features in the same composition is important: for instance, when blending a character's nose to the head (Figure 1(b)) or the leg and tail of an animal to its body (Figure 14 (right)), the junction has to be smooth at the top but sharp at the bottom. To achieve this, our organic opening function is set to be asymmetric: surfaces smoothly blend where they have nearly orthogonal gradients, while a clean union generating sharp creases is used when gradients tend to be in opposite directions (see Figure 14(left)). This enables us to get detailed organic-like models with both smooth parts and sharp creases.

Table I. Parameter values defining our different opening functions.

	α_0	α_1	α_2	θ_0	θ_1	θ_2	w_0	w_1
Camel	0	$\pi/2$	π	$\pi/4$	0	$\pi/4$	1	1
Organic	0	$\pi/3$	$3\pi/4$	$\pi/6$	0	$\pi/4$	3	1
Contact	0	$\pi/2$	π	0	$\pi/10$	$\pi/4$	1	0.7



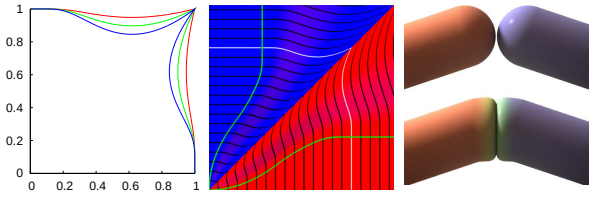
(a) (b) (c)

Fig. 15. Three useful opening functions: (a) camel and (b) organic opening functions used with the blending operator g_b; (c) contact opening function used with the bulge-in-contact operator g_c.

5.2 Bulge in contact for soft objects animation

To illustrate the variety of compositions we can achieve, we designed a "bulge-in-contact" operator g_c, inspired by [Cani 1993], which instead of blending primitives that intersect, mimics the effect of two surfaces in contact that bulge as if made from a soft material such as rubber (Figures 16, 1(c) and (d)).

Keeping in mind that the silhouette curve $\bar{s}$ directly gives the shape of the deformation, Figure 16(a) shows the deformation of f_1 on the right and of f_2 on top. We set $\bar{s}$ such as to create smoothly increasing and then decreasing bulges. This is done by using values smaller than 0.5 on the silhouette curve. Figure 16(c) shows the resulting point of contact (no deformation) and the deformation as f_1 and f_2 overlap.



(a) (b) (c)

Fig. 16. (a) Plots of $\bar{s_c}$ for $K = 0.25$ (red), $K = 0.5$ (green) and $K = 0.75$ (blue). (b) The corresponding operator g_c when $\theta = \frac{\pi}{8}$ and $K = 0.8$. (c) The application of our operator g_c.

Defining these contact silhouette curves is done using the parametric expression $\bar{s}_c(t)(x_c(t), y_c(t))$ with:

$$x_c(t) = \begin{cases} t & \text{if } t \geq 1 \\ 1 - K \exp\left(1 - \frac{1}{2-t}\right)(t-1) & \text{otherwise} \end{cases}$$
$$y_c(t) = \begin{cases} 1 - K \exp\left(1 - \frac{1}{t}\right)(1-t) & \text{if } t \geq 1 \\ 2 - t & \text{otherwise} \end{cases} \quad (31)$$

for $t \in [0,2]$, where $K \in [0,1]$ is a parameter that controls the thickness of the bump. The effect of K on the iso-curves of $\bar{g_c}$ is illustrated in Figure 16 (a). The curve $\bar{s_c}$ is perfectly flat at the extremities and is, as desired, only C^0 continuous at $\varphi = \frac{\pi}{4}$ in order to simulate the fold to be created between the two input surfaces. Again, the polar form $\bar{s_c}(\theta)$ can easily be evaluated using binary search, as done for $\bar{s_b}$ in Section 5.1.

Finally, a *contact opening function*, whose parameter values are given in the third row of Table I, is designed so that when used with our bulge-in-contact operator, objects do not deform at a distance but bulge when they intersect, mimicking contact between soft objects, as illustrated in Figure 15(c). The operator g_c is illustrated with $K = 0.8$ in Figures 1(d) and 16(c).

6. RESULTS AND DISCUSSION

Even though the implementation of the set of functions presented here can be considered complex, its usage is not. The functions can be used in an interactive modeling framework as any other operator; objects are added to the scene, selected and the operator is applied to two of them. With most traditional implicit operators the resulting shape is fixed after applying the operator. In contrast our new set of functions provide ways to further modify the shape, by altering the silhouette curve and the opening function in a graphical editor, however, most of the useful opening functions are defined in this paper and cover most of the practical cases. New opening functions only need be defined for specialized operations.

The usefulness of our generic composition framework for constructive modeling and animation applications is illustrated in Figures 17 and 1(d). The model presented in Figure 17 has been built using ten different primitives combined with either our blending operator together with the camel opening function or with our clean union operator (generating sharp edges). In the case of a blend overlapping a sharp edge, we use the procedure presented in Section 7 and in Figure 19. The object has been created interactively using the Blob-tree [Wyvill et al. 1999] in several sessions over a period of about eight hours. In the "lava lamp" animation, g_c is used in conjunction with our blending operator to also get some blending when bubbles leave the bottom of the lamp. Other interesting animation effects can be easily produced by dynamically modifying the opening function parameters during an animation. For instance, Figure 18 illustrates an animation where primitives do not blend when they are not in contact, but do blend (and thus deform each other) when they separate even though they are not in contact anymore, as in standard blending.

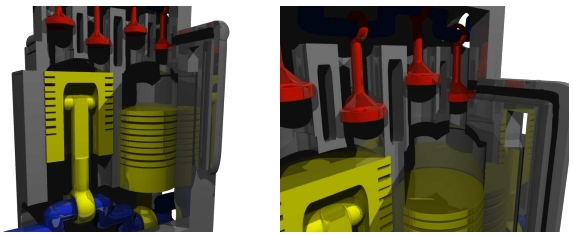


Fig. 17. Constructive modeling examples built using our new operators. In this model, clean union and clean difference operators have been used in conjunction with gradient-based blending operators. Both sharp edges and subsequent gradient-controlled blends have thus been efficiently performed.

Achieving good quality results leads us to use more complex equations for composition than in all previous methods. The efficient evaluation of our operators is based on pre-computations that only need to be performed once for each operator and for each opening function, and just requires the memory space of a few textures. This implementation allows us to use arbitrarily complex expressions for g and θ, while answering field queries at the cost of a simple texture fetch. Therefore, when designing a new operator, one can focus on the desired effect rather than on the best properties a closed form equation could achieve. The precomputation of our operator and its partial derivatives in 128^3 grids takes 600 ms on a Core I7 950 and their transfer to the device memory from the host memory takes 3 ms. The 786 million evaluations of g required for the rendering of Figure 11 take less than 270 ms on a NVIDIA GTX 480.

7. LIMITATIONS

Firstly, as we emphasized in Section 4.1, adequate slope for the opening function has to be used, depending on the slope of the input field functions, for not creating extra artifacts when removing a bulge. In practice, we use the interactive feedback of implicit modeling systems to pre-set the steepness parameters w_0 and w_1 of our opening functions according to the family of field function used. Typically, we need sharper opening functions to blend soft-objects based on smooth polynomial field functions than to compose sharper convolution surfaces. Once the steepness parameters are set, we design specific opening functions by adjusting the three input values for $(\alpha_i, \theta(\alpha_i))$. Providing an automatic tuning of the opening function's slope would be a better solution. The method, as currently implemented, still requires little user interaction since the manual tuning only needs to be done once for a given family of field functions, for each opening function.

A drawback of our gradient-based operators is that they theoretically require at least the continuity of the gradient directions of the field functions f_1 and f_2 to be combined. Using our composition framework in a more general setting, where the field may have a few singular points (with null gradients and therefore undefined angle between gradients) would however be desirable, as well as using it even for shapes with gradient discontinuities, e.g. for primitives with sharp edges. In case of null gradient vectors, we simply decrease the opening angle to $\min(\theta_0, \theta_1, \theta_2)$ when the gradient norm becomes close to null, so that the opening function is constant at the vicinity of the singularity. This only leaves us with the case of artifacts created by discontinuities of the gradient field. This is illustrated by Figure 19(left), where the camel opening function is used for blending in the vicinity of a sharp edge. The problem can be solved in the following way: while primitives are not in contact, an opening function preventing blend is required, and as soon as the primitives intersect, a blend can be progressively performed. This is done easily by manually tuning a few opening function parameters as illustrated in Figure 19. In an animation, the opening function parameters have thus to be pre-set in key primitive positions such as those presented in Figures 19 and then interpolated during the animation.

As with all recent composition methods, the gradient-based operators are restricted to binary compositions of implicit surfaces. This may limit their applicability in situations where many primitives need to be symmetrically connected at once. Generalizing the idea of using gradients into n-ary composition operators would be a topic for future work. Similarly, although they can be used to compose convolution primitives, our operators cannot replace the

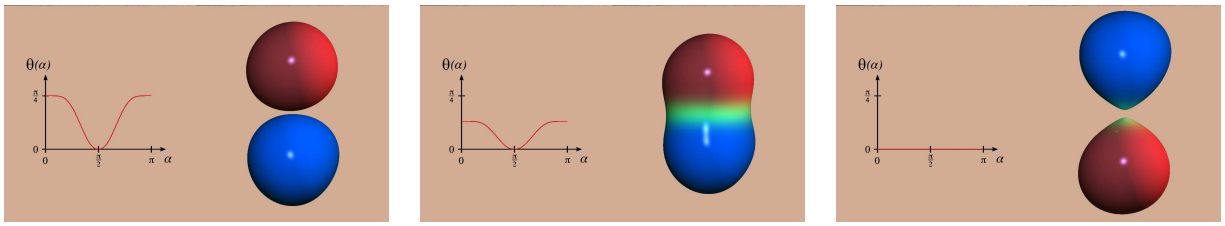
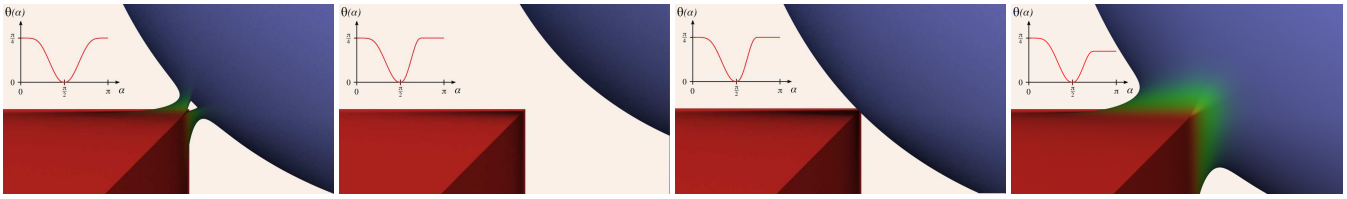


Fig. 18. From left to right, the opening function is modified to create different blending behaviors during an animation.



unwanted handle (camel) clean union while primitives are not in contact blending when primitives intersect

Fig. 19. Composition of a cube and a sphere at a gradient discontinuity (an edge of the cube). On the left, the artifact generated when the camel opening function is used for blending, and then, opening functions solving the problem when the sphere moves and intersects the cube.

sum performing the convolution integral of each element (points, segment, triangles) of a complex skeleton defining a convolution surface [Bloomenthal 1997b]. We do not provide a solution to the problem of a single convolution primitive, from unintentionally blending with itself.

8. CONCLUSION AND FUTURE WORK

We have presented a generic family of composition operators, applicable to both constructive implicit modeling and animation. These operators, implemented in real-time on the GPU, allow precise tuning of the transition between smooth blends and sharp union within a single composition operation, making compositions both more general and more predictable. The main feature of our approach is the use of gradient-controlled blending: our operators are not only a function of the field functions to be combined, but also of their gradient. This avoids the well known weaknesses of implicit modeling, such as creating unwanted bulges and blending at distance. In general, the topology of the resulting shape can be set to the topology of the union. Moreover, our operators naturally prevent small details from blowing up even when blended into much larger primitives, thanks to the sharp changes of gradient values. In consequence, complex models with both smooth parts and sharp creases can be easily created, in contrast with the common idea that implicit surfaces can only represent smooth, blobby shapes.

The natural extension of using the angle between field gradients would be the use of the information on the norm of these gradients to improve blends. This norm could be used to automatically set the slope parameters of the opening functions, according to steepness of the input fields. Another extension would be to rely on the second derivatives of the input fields to detect fast local gradient variations. This would help us to avoid user interaction currently necessary to adjust opening functions in areas where the field functions are very distorted or at the vicinity of gradient discontinuities such as sharp edges. Lastly, the general methodology we developed for the opening functions, i.e. defining $\mathcal{C}^\infty$ curves with some shape control parameters, could be re-used to introduce local-support, $\mathcal{C}^\infty$ fields for implicit primitives. This would ensure always having smooth gradient fields, whatever the number of gradient-based compositions.

9. ACKNOWLEDGMENTS

This work has been partially funded by the IM&M project (ANR-11-JS02-007), the Natural Science and Engineering Research Council, Canada, and by the advanced grant *EXPRESSIVE* from the European Research council.

REFERENCES

ALEXE, A., BARTHE, L., CANI, M., AND GAILDRAT, V. 2005. Shape modelling by sketching using convolution surfaces. In *Pacific Graphics (Short Papers)*.

BARTHE, L., DODGSON, N. A., SABIN, M. A., WYVILL, B., AND GAILDRAT, V. 2003. Two-dimensional potential fields for advanced implicit modeling operators. *Computer Graphics Forum 22,* 1, 23–33.

BARTHE, L., GAILDRAT, V., AND CAUBET, R. 2001. Extrusion of 1d implicit profiles: Theory and first application. *International Journal of Shape Modeling 7*, 179–199.

BARTHE, L., WYVILL, B., AND DE GROOT, E. 2004. Controllable binary csg operators for "soft objects". *International Journal of Shape Modeling 10,* 2, 135–154.

BERNHARDT, A., BARTHE, L., CANI, M.-P., AND WYVILL, B. 2010. Implicit blending revisited. *Proc. of Eurographics, Computer Graphics Forum 29,* 2, 367–376.

BERNHARDT, A., PIHUIT, A., CANI, M. P., AND BARTHE, L. 2008. Matisse : Painting 2d regions for modeling free-form shapes. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*. 57–64.

BLINN, J. F. 1982. A generalization of algebraic surface drawing. *ACM Trans. Graph. 1,* 3, 235–256.

BLOOMENTHAL, J. 1997a. Bulge elimination in convolution surfaces. In *Computer Graphics Forum*. Vol. 16. 31–41.

BLOOMENTHAL, J., Ed. 1997b. *Introduction to Implicit Surfaces*. Morgan Kaufmann.

BRAZIL, E., MACEDO, I., SOUSA, M. C., DE FIGUEIREDO, L., AND VELHO, L. 2010. Sketching variational hermite-rbf implicits. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*. SBIM '10. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 1–8.

CANI, M.-P. 1993. An implicit formulation for precise contact modeling between flexible solids. In *20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1993*. 313–320. Published as Marie-Paule Gascuel.

HOFFMANN, C. AND HOPCROFT, J. 1985. Automatic surface generation in computer aided design. *The Visual Computer 1,* 2, 92–100.

HSU, P. C. AND LEE, C. 2003. Field functions for blending range controls on soft objects. *Proc. of Eurographics, Computer Graphics Forum 22,* 3, 233–242.

KARPENKO, O., HUGHES, J., AND RASKAR, R. 2002. Free-form sketching with varational implicit surfaces. *Proc. of Eurographics 2002 21*, 585–594.

PASKO, A. AND ADZHIEV, V. 2004. Function-based shape modeling: mathematical framework and specialized language. In *Automated Deduction in Geometry, Lecture Notes in Artificial Intelligence 2930*. 132–160.

PASKO, A., ADZHIEV, V., SOURIN, A., AND SAVCHENKO, V. 1995. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer 11,* 8, 429–446.

PASKO, G. I., PASKO, A. A., AND KUNII, T. L. 2005. Bounded blending for Function-Based shape modeling. *IEEE Comput. Graph. Appl. 25,* 2, 36–45.

RICCI, A. 1973. A Constructive Geometry for Computer Graphics. *computer journal 16,* 2 (May), 157–160.

ROCKWOOD, A. 1989. The displacement method for implicit blending surfaces in solid models. *ACM Transactions on Graphics 8,* 4, 279–297.

SABIN, M.-A. 1968. The use of potential surfaces for numerical geometry. In *Tech. Report VTO/MS/153, British Aerospace Corp., Weybridge, U.K.*

SAVCHENKO, V., PASKO, A., OKUNEV, O., AND KUNII, T. 1995. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum 14*, 181–188.

TAI, C., ZHANG, H., AND FONG, J. 2004. Prototype modeling from sketched silhouettes based on convolution surfaces. In *Computer Graphics Forum*. Vol. 23. 71–83.

WYVILL, B., FOSTER, K., JEPP, P., SCHMIDT, R., SOUSA, M. C., AND JORGE, J. 2005. Sketch based construction and rendering of implicit models. In *Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging*. 67–74.

WYVILL, B., GUY, A., AND GALIN, E. 1999. Extending the csg tree - warping, blending and boolean operations in an implicit surface modeling system. *Comput. Graph. Forum 18,* 2, 149–158.

WYVILL, G., MCPHEETERS, C., AND WYVILL, B. 1986. Data Structure for Soft Objects. *The Visual Computer 2,* 4 (February), 227–234.